POWER AMPLIFIER PRE-DISTORTION

TECHNICAL FIELD

The present invention relates to digital pre-distortion in power amplifiers with memory effects.

BACKGROUND

Power amplifiers are known to add more or less distortion to the signal they are designed to amplify. The reason for this is that a power amplifier has a non-linear input-output signal characteristic. This shows up as a broadened spectrum around the desired amplified signal, and as an unwanted inband component of the signal. As a counter-measure to decrease the effects of non-linearity, it is known to pre-distort the signal at the input of the amplifier as to give an un-distorted amplified signal at the output of the amplifier. This technique is called pre-distortion. Pre-distortion as implemented today normally uses a look-up table that is used to multiply the signal. The entry into the table is the magnitude of the signal at every time sample.

Memory effects is another problem related to power amplifiers. Memory effects typically show up as a non-symmetrical spectrum around the carrier at the output of a power amplifier. That is, although the carrier (desired signal) spectrum is perfectly symmetrical, the spurious spectrum coming from the distortion may be non-symmetrical with respect to the center of the carrier.

The methods commonly used to handle non-linearity do not in general take into account memory effects of the power amplifier. As the term "memory effects" indicates, there is a dependence not only on the present sample but also on delayed samples of the signal. Thus, a single table approach cannot take care of memory effects, but can only handle non-linearity.

Lei Ding et. al. [1] inspired by work by Kim and Konstantinou [2] have derived a pre-distortion method based on what they call "Memory Polynomials" that very well model memory effects. However, a drawback of this method is that it requires recalculation of the memory polynomials for each new input signal amplitude, which can be computationally costly, especially if many polynomials of high order are used. A further drawback of this method is the computationally costly training procedure used to determine the polynomial coefficients.

SUMMARY

An object of the present invention is to provide a computationally efficient training method for pre-distortion based on memory polynomials.

This object is achieved in accordance with the attached claims.

Briefly, the present invention is based on a FIR filter structure including individual look-up tables for the filter taps, where each look-up table represents a discretized memory polynomial. Assuming two look-up tables, the training method in accordance with the invention is based on the observation that the look-up table that compensates for memory effects typically has much smaller elements than the look-up table that compensates for non-linearities. This makes it possible to easily determine a first approximation of this dominating look-up table by simply neglecting the other table. This approximation can in turn be used to determine a first approximation of the table compensating for the memory. This procedure can then be iterated by using the latest approximation of one table for determining a refined approximation of the other table until convergence is reached. An advantage of this method is that in each step only a single table has to be determined (the other table is assumed to be constant), which is a much simpler process than to determine both tables simultaneously in a single step.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further objects and advantages thereof, may best be understood by making reference to the following description taken together with the accompanying drawings, in which:

Fig. 1 is a diagram illustrating the non-linear input-output signal characteristic of a power amplifier;

Fig. 2 is a diagram illustrating the spectrum of the signal amplified by a non-linear power amplifier;

Fig. 3 is a diagram illustrating the input-output signal characteristic of a power amplifier pre-distorter for removing the non-linearity in Fig. 1;

Fig. 4 is a diagram illustrating the input-output signal characteristic of a power amplifier provided with pre-distortion;

Fig. 5 is a diagram illustrating the spectrum of the signal amplified by a non-linear power amplifier with memory;

Fig. 6 is a diagram illustrating discretization of polynomials in accordance with the present invention;

Fig. 7 is a block diagram of an exemplary embodiment of a pre-distorter suitable to be trained in accordance with the present invention;

Fig. 8 is a block diagram of an exemplary embodiment of a base station including a power amplifier provided with a pre-distorter in accordance with Fig. 7; and

Fig. 9 is a flow chart illustrating an exemplary embodiment of the training method in accordance with the present invention.

DETAILED DESCRIPTION

In the following description the same reference designations will be used for the same or similar elements throughout the figures of the drawings.

Before the invention is described in detail, a brief description of the underlying problem will be given below.

Fig. 1 illustrates the non-linear input-output signal characteristic of a power amplifier. At low input signal amplitudes the amplifier is almost linear, but at higher amplitudes it becomes more and more non-linear until it is saturated. This non-linearity shows up as a broadened spectrum around the desired amplified signal (and as an unwanted inband component of the signal), as illustrated in Fig. 2. As a counter-measure to decrease the effects of non-linearity, it is known to pre-distort the signal at the input of the amplifier to give an un-distorted amplified signal at the output of the amplifier. This technique is called pre-distortion and is illustrated in Fig. 3. The input-output signal characteristic for a pre-distorted power amplifier is essentially linear up to saturation, as illustrated in Fig. 4.

Memory effects is another problem related to power amplifiers. Memory effects typically show up as a non-symmetrical spectrum around the carrier at the output of a power amplifier, as illustrated in Fig. 5.. That is, although the carrier (desired signal) spectrum is perfectly symmetrical, the spurious spectrum coming from the distortion may be non-symmetrical with respect to the center of the carrier.

There is a theoretical way of designing a pre-distorter that takes care of all memory effects. This is called the Volterra series. The Volterra series is an extension to the well-known Taylor series, which can be used as a pre-distorter for memory-less amplifiers. The Volterra series, however, also takes into account time-delayed terms that may quite accurately model the pre-distortion, and may therefore be used to suppress the distortion spectrum. However, a Volterra series quite rapidly gets large in terms of the number of possible terms in the expansion. For example, a polynomial of degree 5 with a memory depth (maximum delay) of 5 sample units will give rise to at least 500 coefficients.

Since the full Volterra series can not be implemented with reasonable complexity, an approximation based on "Memory Polynomials" has been suggested in [1]. In this approximation the pre-distortion *PD(n)* may be expressed as:

$$PD(n) = \sum_{k=1}^{K} \sum_{q=0}^{Q} a_{kq} x(n-q) |x(n-q)|^{k-1} \tag{1}$$

Unfortunately this expression is still quite complicated, and a drawback of this prior art method is that the expression has to be evaluated for each new input sample *x(n)*. However, as will be shown below, this expression may be rewritten into a more suitable form for practical implementation. The derivation essentially includes three steps:

1. Separate the double sum into partial sums including only terms with the same delay. This gives:

$$PD(n) = \sum_{k=1}^{K} \sum_{q=0}^{Q} a_{kq} x(n-q) |x(n-q)|^{k-1} =$$

$$\begin{array}{c}\text{Separate}\\\text{delays}\\\text{into}\\\text{different}\\\text{sums}\end{array}\left\{\begin{array}{l} = \sum_{k=1}^{K} a_{k0} x(n) |x(n)|^{k-1} + \\ + \sum_{k=1}^{K} a_{k1} x(n-1) |x(n-1)|^{k-1} + \\ \vdots \\ + \sum_{k=1}^{K} a_{kQ} x(n-Q) |x(n-Q)|^{k-1} \end{array}\right.$$

2. Here it is noted that the delayed signals *x(n-q)* do not depend on the summation indices *k*. Thus the partial sums may be factorized into:

$$\left.\begin{array}{c}\text{Factorize}\\ k \text{ and } q\\ \text{dependence}\\ \text{in}\\ \text{each}\\ \text{sum}\end{array}\right\}\left\{\begin{aligned} &= x(n)\underbrace{\sum_{k=1}^{K} a_{k0}|x(n)|^{k-1}}_{T_0(|x(n)|)} + \\ &+ x(n-1)\underbrace{\sum_{k=1}^{K} a_{k1}|x(n-1)|^{k-1}}_{T_1(|x(n-1)|)} + \\ &\vdots \\ &+ x(n-Q)\underbrace{\sum_{k=1}^{K} a_{kQ}|x(n-Q)|^{k-1}}_{T_Q(|x(n-Q)|)}\end{aligned}\right.$$

3. Identify the polynomials $T_q(|x(n-q)|)$ to obtain:

$$PD(n) = \sum_{q=0}^{Q} x(n-q)T_q(|x(n-q)|) \tag{2}$$

It is noted in (2) that $T_q(|x(n-q)|)$ are polynomials in the absolute value of the (complex) variable $x(n-q)$. Thus, by multiplying each delayed complex sample $x(n-q)$ by a polynomial in $|x(n-q)|$ (which has the same delay q) and summing up the products for all delays q, the same end result $P(n)$ as in [3] will be obtained. However, this new approach has the advantage that the polynomials T_q may be sampled at appropriate values of $|x(n-q)|$, as illustrated in Fig. 6, and stored in look-up tables. This will reduce the pre-distorter to a simple FIR filter structure, in which the normally constant filter coefficients are replaced by these look-up tables, as illustrated in Fig. 7. The problem addressed by the present invention is to determine the values of the polynomials T_q at the sampling points in Fig. 6.

In the exemplary embodiment of the present invention illustrated in Fig. 7, the complex input signal $x(n)$ is forwarded to an absolute value block 10 and to a multiplier 12. The absolute value signal from block 10 is forwarded to a look-up table LUT0 representing a sampled version of polynomial T_0. The corresponding (generally complex) value from look-up table LUT0 is forwarded to multiplier 12, where it multiplies the input signal sample $x(n)$. Input signal $x(n)$ is also forwarded to a delay block D, where it is delayed one or several sample periods for forming a delayed sample $x(n-1)$. This delayed sample is processed in the same way as the non-delayed sample by an absolute value block 10, a multiplier 12 and a look-up table LUT1. However, look-up table LUT1 now represents a sampled version of polynomial T_1 instead of T_0. As illustrated in Fig. 7, further delays and look-up tables may be included. Finally, the obtained products are added to each other in adders 14 to form the pre-distorted signal $PD(n)$. Look-up tables used in accordance with the present invention make computation in real time much more efficient than the polynomial computation for each sample of the input signal used in [1]. The look-up tables may be updated (by using the training method described below) to keep track of slow changes in the characteristics of the power amplifier.

Fig. 8 is a block diagram of an exemplary embodiment of a base station including a power amplifier provided with a pre-distorter in accordance with the present invention. In Fig. 8 elements that are not necessary for understanding the invention have been omitted. The baseband complex signal $x(n)$ is forwarded to a pre-distorter 30 in accordance with the present invention. The pre-distorted signal $y(n)$ is up-converted to intermediate frequency (IF) in a digital up-converter 32 and converted into an analog signal in a D/A converter 34, which in turn is up-converted to radio frequency (RF) by an analog up-converter 36. The RF signal is forwarded to a power amplifier 38, and the amplified signal is forwarded to an antenna. The amplified RF signal is also forwarded to a feedback down-conversion chain including an analog down-converter 40, an A/D converter 42 and a digital down-converter 44. The down-converted feedback signal $z(n)$ is forwarded to a trainer 46, which also receives the pre-distorted input signal $y(n)$ for determining the look-up tables in pre-distorter 30 in accordance with the mathematical principles described below.

The described look-up table based pre-distorter may be implemented as an FPGA (Field Programmable Gate Array). Another possibility is to use a micro processor or a micro/signal processor combination and corresponding software. The actual computation of the look-up table entries may be done in an off-line manner at a slow update speed, as described below.

The training method performed by trainer 46 in Fig. 8 will now be described in more detail. In order to illustrate the training process, a pre-distorter 30 including two look-up tables T_0 and T_1 (corresponding to LUT0 and LUT1 in fig. 7, with a one sample delay as an example) will be assumed. With this assumption equation (2) becomes:

$$y(n) = x(n) \cdot T_0(|x(n)|) + x(n-1) \cdot T_1(|x(n-1)|) \tag{3}$$

A training procedure may be implemented by noting that *z(n)* should be equal to *y(n)* if the distorting action of the power amplifier is undone, i.e. if the same pre-distortion is applied to *z(n)*. Thus, the following equation may be used as a basis for a training procedure for determining the look-up tables T_0 and T_1:

$$z(n) \cdot T_0(|z(n)|) + z(n-1) \cdot T_1(|z(n-1)|) = y(n) \tag{4}$$

The training procedure is based on measuring a large batch of signal pairs

$$(y(i), z(i)), \quad i = 1, \ldots, N$$

where N is the number of signal pairs in the batch (depending on the required accuracy, the number of table entries etc, *N* is typically between 2000 and 50 000). Since the samples in the batch should fulfill equation (4), one obtains a system of equations. Using traditional methods to solve such a system of equations involves approximating the look-up tables by polynomials in $|z(n)|$ and $|z(n-1)|$, respectively, and solving the resulting approximate equations by the method of least mean squares by solving the so called normal equations [1]. This method, however, involves rather extensive numerical computations, especially since complex numbers and complex matrix inversions are involved in the process.

The present invention suggests a simpler method for determining look-up tables from equation (4). This method is based on the observation that table T_1, which takes care of memory effects, typically has elements that are at least an order of magnitude smaller than the elements of table T_0. Thus, as a first approximation equation (4) may be written as:

$$z(n) \cdot T_0(|z(n)|) = y(n) \tag{5}$$

This equation is much easier to solve. Since equation (5) has the same form as a simple pre-distorter without memory effect compensation, any method used for determining the look-up table of such a pre-distorter may be used to solve it. One such method may, for example, be found by dividing the range of $|z(n)|$ into intervals, and then solve equation (5) for each amplitude level. Since there is a multitude of data for each amplitude level, one might compute the average of all the data belonging to each amplitude level k to obtain a first approximation $T_0^{(1)}$ of T_0 in accordance with the formula:

$$T_0^{(1)}(k) = \frac{1}{N_k} \cdot \sum_{i=[\text{index}]} \frac{y(i)}{z(i)} \tag{6}$$

where the vector "[index]" includes the indices of the signal samples in the measured batch that lie in the interval that corresponds to table index k and N_k represents the number of components of this vector. A standard search algorithm (not described here) may be used for this purpose. Another method of determining table T_0 from equation (5) is described in [3].

Once the first approximation $T_0^{(1)}$ has been found, this approximation may be used in equation (4) to determine a first approximation of *T*1. Moving the first term to the right hand side gives:

$$z(n-1)\cdot T_1(|z(n-1)|)=y(n)-z(n)\cdot T_0^{(1)}(|z(n)|)=w(n-1) \qquad (7)$$

Since the index name *n* is arbitrary (it does not matter whether it is called *n* or *n*-1), it is noted that equation (7) has the same form as equation (5). This fact may be used to find a first approximation $T_1^{(1)}$ of T_1 by using the same method as for *T*0. Thus, $T_1^{(1)}$ may, for example, be determined as the average:

$$T_1^{(1)}(k)=\frac{1}{N_k}\cdot\sum_{i=[\text{index}]}\frac{w(i)}{z(i)} \qquad (8)$$

In a simple form of the invention, the approximations $T_0^{(1)}$ and $T_1^{(1)}$ may be used directly as the output of the training procedure. However, further refinements may be achieved by repeating the "trick" in equation (7) one or several times. Thus, to obtain a second approximation $T_0^{(2)}$ of T_0, the first approximation $T_1^{(1)}$ of T_1 is inserted into equation (4). Once again an equation having the same form as equation (5) is obtained. This equation may be solved approximately using, for example, the averaging method described above. A second approximation $T_1^{(2)}$ of T_1 may be obtained in a similar way. This process may be repeated to obtain approximations of higher orders until convergence is reached (i.e. until there are only insignificant changes to the approximations from one iteration to the next). Depending on he required accuracy, convergence is usually obtained after 3-5 iterations. However, if the accuracy requirements are relaxed, acceptable tables may even be obtained after the first or second iteration.

Thus, by solving equation (4) iteratively as described above, the training problem has been reduced to repeatedly solving equations having the same form as equation (5), which is only a single table problem. It is stressed that any method suitable for solving such a single table problem may be used to find the successive approximations of T_0 and T_1.

Equation (4) is valid both for a first time training and for updating the look-up tables. The method described above assumed that the tables had not been determined yet. For this reason T_1 was initially set to 0. However, if the tables are updated, the current table T_1 may be used as an initial guess instead, since it is probably closer to the correct table.

Another variation of the method described above is to reverse the roles of tables T_0 and T_1 by initially setting all elements of table T_0 to 1 (instead of setting the elements of T_1 to 0) and determine $T_1^{(1)}$ before $T_0^{(1)}$. A similar approach for updating the tables uses the current table T_0 as an initial guess instead of the current table T_1.

Fig. 9 is a flow chart illustrating an exemplary embodiment of the training method in accordance with the present invention. Step S1 sets the elements of table T_1 to predetermined values, typically 0 for a first time training and the current T_1 values for an update. Then T_0 is estimated by solving equation (6). Step S2 sets T_0 to the determined estimate in equation (4) for obtaining an equation having the same form as equation (5), This equation is solved for an estimate of T_1. If simplicity is more important than accuracy, it is possible to stop here and output the obtained estimates as the final tables. However, preferably a few more iterations (steps S3-S5) are performed. Step S3 sets T_1 to the latest determined estimate of T_1. Then the estimate of T_0 is refined by solving an equation similar to equation (5). Step S4 sets T_0 to the latest determined estimate of T_0. Then the estimate of T_1 is refined by solving an equation similar to equation (5). (Another simple embodiment may be obtained by stopping after step S3. This embodiment involves two iterations for T_0, but only one iteration for T_1. This embodiment may, for example, suffice for an update, since a rather good estimate is already available for T_1 in this case.) Step S5 tests whether the tables have converged. This can be done, for example, by summing the absolute values (or squares) of the differences between corresponding elements of the current estimates and the previous estimates and testing whether the obtained sum is smaller than a predetermined threshold. If the tables have converged, the current estimates are provided as the final tables in step S6. If the tables have not converged, steps S3-S5 are repeated. Typically this method is implemented by a micro processor or a micro/signal processor combination and corresponding software.

The outlined iterative method of successively improving table accuracy may be extended to include more tables. For example, a third table, which takes into account memory effects of another time delay, may be included in the same manner as in the two-table solution. The iteration may be outlined as follows: Perform the iteration scheme as outlined above until convergence is reached for tables T_0 and T_1. Then add a third table T_2 corresponding to another time delay and determine an estimate of this table using the equation:

$$z(n-2)\cdot T_2(|z(n-2)|)=y(n)-z(n)\cdot T_0(|z(n)|)-z(n-1)\cdot T_1(|z(n-1)|)=v(n-2)$$

which again has the same form as equation (5). Further iterations may now be performed using estimates of two tables to determine a refined estimate of the third table until convergence is reached. A variation of this scheme is to estimate the third table as soon as the first estimates of the two other tables have been obtained. The described extension gives best results if the third table has elements with significantly smaller magnitudes than the second table.

The merits of this invention are three-fold: Firstly, already implemented single-table algorithms may be used for both the first table and the second memory table (and possibly further tables). Secondly, an iterative method may be used which usually is very attractive to implement in software due to its simple structure. Thirdly, exhausting computations involving inversion of complex-valued matrices and matrix-matrix multiplications as in the method of Least Mean Squares can be avoided entirely. The present disclosure makes it possible to implement multi-table calculations in processors of the same size as used for single-tables. The execution time will only be slightly longer than for a single-table. Usually only a few loops in the iteration scheme will be necessary to obtain the same convergence as the Least-mean Squares algorithm.

It will be understood by those skilled in the art that various modifications and changes may be made to the present invention without departure from the scope thereof, which is defined by the appended claims.